DOCUMENT RESUME

ED 371 649 HE 027 490

AUTHOR Bezeau, Lawrence M.
TITLE Timetabling an Academic Department with Linear Programming.
PUB DATE 30 May 93
NOTE 23p.; Paper presented at the Conference of the Canadian Association for the Study of Educational Administration (Ottawa, Ontario, Canada, June 13, 1993).
PUB TYPE Reports - Descriptive (141) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS Computer Oriented Programs; *Departments; Foreign Countries; Higher Education; *Linear Programming; Operations Research; *Scheduling; *School Schedules; Secondary Education; *Universities
IDENTIFIERS Mathematical Programing; *University of New Brunswick (Canada)

ABSTRACT

This paper describes an approach to faculty timetabling and course scheduling that uses computerized linear programming. After reviewing the literature on linear programming, the paper discusses the process whereby a timetable was created for a department at the University of New Brunswick. Faculty were surveyed with respect to course offerings and preferred days and times of instruction. Variables were then constructed and data entered using the IBM Mathematical Programming System (MPS) on a mainframe computer. For a department of 13 faculty members the result was a set of 200 variables, about 35 of which were to be selected through linear programming to form the basis for determining feasible and optimal section offerings for the timetabling period. Several timetables and lists were prepared for faculty members to review, with their comments used to adjust the program to produce the most acceptable results for the department as a whole. Application of this process at the university and secondary school level is discussed. (Contains 30 references.) (MDM)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED 371 649

Timetabling an Academic Department with Linear Programming

by

Lawrence M. Bezeau
New Brunswick Centre for Educational Administration
University of New Brunswick
Post Office Box 45333
Fredericton, Canada E3B 6E3

(BEZEAU@UNB.CA)

1993 05 30

Table of Contents

Definitions and Examples 2
Review of the Literature 6
The Timetabling Problem 8
Linear Programming 10
Variable Construction 11
Constraint Coding 13
Objective Function Specification 16
Input File Preparation 16
Ad Hoc Adjustments 18
Linear Programming Output 18
Application to Conflict Matrices 19
Conclusions 20
Bibliography 21

HE 027 490

BEST COPY AVAILABLE

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
☑ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Lawrence M. Bezeau

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

Timetabling an Academic Department with Linear Programming

This paper is a description of an approach to the faculty timetabling problem using linear programming based on my experience of timetabling a small university department in this manner for three years. It is practical in the sense that it demonstrates the efficiency and effectiveness of linear programming as a means of university timetabling and shows how the variables and constraints of the linear program can be conceptualized to solve the timetabling problem.

Linear programming involves the optimization (maximization or minimization) of a linear objective function subject to linear constraints. It has been widely used in government and industry to solve problems associated with scheduling, transportation, construction, and manufacturing. It has also been used before in timetabling, as the review of the literature below indicates. Objections to using linear programming in timetabling have generally been based on concerns about the size of the problem but these are becoming increasingly irrelevant as computers become more powerful and as computer time becomes less expensive.

Definitions and Examples

The following paragraphs contain some commonly used timetabling terms and their definitions. In practice many of these terms are used loosely and have general or multiple meanings. Because this exposition requires a set of precise and consistent terms, I have given these words and phrases exact definitions but, in doing so, have attempted to reflect their common usage. Some illustrative examples are given for many of the defined terms. These examples do not exhaust current practice and, above all, do not exhaust the possibilities.

Course

The word "course" is widely understood to mean a discrete administrative unit of instruction. In any institution, courses are identified by a unique course number which often includes a non-numerical prefix as well as a numeral, and by a title which may not be unique. In fact it is the smallest unit of instruction for which a student can obtain credit. One does not receive credit for a fraction of a course and only rarely are students allowed to obtain credit for a single course more than once. Courses can be contrasted with larger administrative units of instruction such as programs.

Courses must also be distinguished from their particular offerings, called course sections.

Course Section

A course section is a particular offering of a course with a certain group of students during a definite period of calendar time. Course sections within a given period of time usually have a unique identifying designation in addition to the course number. This may be called a section number. A course section is the smallest unit of instruction that a student either takes or does not take. Multiple sections of the same course can be offered during the same calendar period but students almost never register concurrently for more than one section of the same course. At the secondary school level the terms "singleton", "doubleton", and "tripleton" are used to refer to courses for which one, two, and three sections respectively are offered during the same period of calendar time. For a given course section, classes are usually held in the same location with the same instructor; but the defining characteristic is probably the group of students who are registered in the section, rather than the classroom, instructor, or time of offering.

Timetabling Period

This is the period of calendar time for which a course timetable will apply and is normally equal to the calendar length of course sections. In universities these include academic years, semesters, quarters, and terms. Secondary schools generally use school years or semesters. A number of less standardized timetabling periods may also be employed such as summer sessions, extended summer sessions, and intersessions. Most timetabling periods in current use in education are between two weeks and ten months in length.

There are some scheduling problems that cross timetabling periods but these usually involve the choice of courses rather than the times at which chosen courses are offered. Sequenced courses must be offered so that pre-requisites come before the courses that require them. Some courses must be offered in every timetabling period, perhaps in multiple sections, whereas others must only be offered once in every second or third timetabling period if sufficient enrolments are to be maintained. The availability of faculty members and other resources may also influence the choice of course sections to be offered in any given timetabling period.

Section Cycle Period

Timetabling periods are divided into shorter periods that are timetabled repetitively. These are here called "course section cycle periods" or just "cycle periods". At the college and university level the almost universal cycle period during the academic year is the standard five day work week, although some institutions offer classes on Saturdays. At the secondary level there are a number of popular variations. These include daily timetabling and cycle periods of either four or six days both incorporated into the standard five day week. The individual days in four and six day cycles do not correspond to the named days of the standard week and are usually given numbers instead. The purpose of these variations is to avoid the time penalties often borne by course sections offered on Mondays and Fridays. By deliberately allowing the cycle period length to differ from that of the standard week, school officials can be assured that Monday and Friday will not correspond to any one day in the cycle period and that no day in the cycle will be systematically disadvantaged.

Class Period

A class period or just a class is a continuous length of time during which instruction is offered. At the secondary and post-secondary levels these can vary from twenty minutes to four hours or more. Typical daytime courses at post secondary institutions have class periods with nominal lengths of either 1.0 or 1.5 hours. The nominal class period includes a time to change classes of 10 or 15 minutes in these two cases leaving 50 and 75 minutes of actual instruction respectively. Typical evening class periods are of 3.0 nominal hours but one or more breaks during the class leaves about 2.5 hours of actual instruction. Thus actual instruction consumes 83 percent, more or less, of the nominal class period.

At the secondary level, timetables with variable-length class periods are sometimes built on a time period module that is the highest common factor of the class period lengths. Class periods of 40, 60, and 80 minutes would be built from a module of 20 minutes. The modules are timetabled with fixed time boundaries so that class periods can be slotted into the day without unnecessary overlap and without time gaps that cannot be utilized.

Section Cycle Pattern

Within each course section cycle period there is usually a standard predefined pattern of times at which classes begin and end, a set of standard class lengths, and a standard distribution of class periods for given course

sections over the days of the course section cycle period. This is the section cycle pattern. For example, universities that use a five-day weekly cycle period often timetable all class periods for a given course section on Mondays, Wednesdays, and Fridays at the same time or on Tuesdays and Thursdays at the same time. Courses offered on three days have nominal class period lengths of 1.0 hours and those offered on two days have nominal class period lengths of 1.5 hours. All courses will begin on an hour or half-hour boundary. Colleges and universities often follow different patterns than this during timetabling periods outside the usual academic year. The simplest section cycle patterns occur with daily cycle periods for which class periods begin at the same fixed times each day. Fixed daily starting times may also be used with longer cycle periods. The practice of having all classes of a given course section begin at the same time of day is adhered to less frequently at the secondary level than at the post secondary level. This provides greater flexibility and greater complexity in that the timetabling problem involves distributing course sections throughout 40 weekly class periods, for example, rather than five to ten daily class periods. Section cycle patterns are not absolutely necessary for timetabling but are used almost universally to avoid timetable conflicts and to allow for the efficient use of intensively utilized resources.

Section Class Pattern

The section class pattern or just class pattern is the set of class times and durations for a single section of a given course. Some examples include Monday, Wednesday, and Friday at 1030 to 1130; Tuesday at 0900 for 1.5 hours and Wednesday at 1400 for 1.5 hours; and Thursday evening at 1800 for three hours. The term "time slot" is sometimes used as a synonym but this can be misleading since the class periods in the section pattern need not all be at the same time, as the second example above indicates. The set of all section class patterns for a given timetable is, of course, the section cycle pattern.

Class-teacher timetabling problem

This term appears in the theoretical literature more than in practice but it does appear frequently enough to justify a definition. The class-teacher timetabling problem assumes a set of class periods in a section cycle period and a collection of teachers and student groups, not necessarily equal in number. During the section cycle period, each teacher must meet each student group for a specified number of class periods such that no teacher meets more than one student group during any single class period and such that no student group meets more than one teacher during any single class

period. The matching of student groups and teachers during class periods is one-to-one but there is no necessary requirement that either teachers or student groups meet during all class periods. An important simplifying characteristic of this problem is that student groups do not change their membership. This eliminates individual course section conflicts.

Review of the Literature

The literature in the area of school timetabling is uneven and not uniformly helpful. Much of it consists of discussions of the characteristics or advantages of certain types of timetables, an area of concern excluded from this article. The following review of the literature considers only material on the techniques of timetabling and concentrates on linear programming.

Much of the literature on timetabling dates from the 1960s and 1970s when it first became clear that computers could be useful in timetabling. Less has been done since that time. There is a stream of literature dealing with theoretical aspects and computerized solutions found in periodicals in computer science and operations research and another stream consisting of "how to" manuals for practitioners. The books on timetabling, all in the how-to category, (Brookes, 1980; Delacour, 1971; Dempsey and Traverso, 1983; School Timetabling, 1976; Simper, 1980; and Walton, 1972) were published in the United Kingdom except for Dempsey and Traverso. This illustrates another trend, namely the greater interest in this problem in Europe than in North America.

The computer-science operations-research approaches culminated with an extensive review of the literature by Schmidt and Ströhlein in 1980. Of their 343 references, 274 were in English, 57 in German, and 12 in various other European languages. Schmidt and Ströhlein made several predictions of future trends that we can look back on after more than a decade of additional experience. They predicted that (309):

> A major evolution will, therefore, come in the near future. Timetable programs will probably move from remote handling in huge computing centres to minicomputer systems owned by the school and handled directly by the teachers.

This did not happen in the near future but is happening now with microcomputers rather than minicomputers. Schmidt and Ströhlein also predicted that: "... software support by database systems for bookkeeping will grow rapidly" (309). This has turned out to be more true than they perhaps

expected. Many modern school timetabling systems do not actually timetable. Instead they provide data input facilities, arrange the data to make timetabling easy, and produce reports; in other words, only the bookkeeping services envisioned by Schmidt and Ströhlein. This greatly eases the often enormous clerical burden on timetablers.

Shortly before the Schmidt and Ströhlein review, McKillop (1978) had classified techniques for timetabling into four categories: heuristic, combinatorial, graph theoretical, and linear programming. Schmidt and Ströhlein did not present an exhaustive and mutually exclusive system of approaches as did McKillop, but they did discuss the "operations research approach" which consisted essentially of linear programming. They were not optimistic about the future of linear programming in timetabling because the problems can become large and complex very quickly and can consume large amounts of computer time. This, it turns out, is also true of the other approaches. The consumption of large amounts of computer time is no longer very relevant because costs have gone down by several orders of magnitude and continue to decline. What has become important in its place is the effort required to prepare the input for linear programming if the timetabling problem is large.

Lawrie (1969), in an early application of integer linear programming, used what he called "arrangements" as variables. Arrangements consisted of sets of courses for various streams in a school which could be offered at the same times without producing conflicts. This was done in the context of a section cycle pattern consisting simply of a five-day school week divided into 40 equal non-overlapping class periods without any further structure. The problem of avoiding offering two courses at the same time that were to be taken by the same persons had to be solved beforehand to generate the arrangements. The only linear programming constraints in Lawrie's approach prevented more courses in a given subject from being offered in any class period than there were staff members qualified to teach in that subject area. Lawrie originally minimized the number of different arrangements as an objective function but found that this departed considerably from what school principals regarded as optimal. As a consequence he abandoned objective functions in favor of generating a number of feasible solutions that principals could choose from. Lawrie's model, particularly his use of arrangements, raises some important issues. Developing the arrangements, which he did by hand, solves most of what we now regard as the timetabling problem. The process of producing arrangements by hand, besides being very difficult and time consuming, would not likely result in the generation of all possible unique arrangements. Some feasible and perhaps highly desirable arrangements would never make it as

far as the linear program and could never be considered for implementation. This along with the very limited role for the computer prevented Lawrie's model from being followed up. One must also question the highly unstructured section cycle pattern he used, although this may not have been his choice. Even though his course sections all had multiple class periods during the weekly section cycle period, there were no section class patterns. In theory, a five-period mathematics course section could have met for five consecutive class periods during the same day. Such an outcome must be considered very suboptimal.

In a second article describing an approach based on linear programming (Tripathy, 1984), 900 course sections in 25 different areas of specialization at the university graduate level were timetabled by grouping courses into larger groups of courses in which enrolment was required of most of an identifiable group of students. Unlike many other approaches to this problem, Tripathy included rooms in his formulation but grouped them into five categories according to capacity. Each course group was associated with a room category based on expected enrolment. The constraint sets related to the required periods per week, the availability of rooms, and the requirements that the students in a group take certain courses which therefore could not be offered at the same times. The objective function had coefficients based on desirabilities broadly defined. Much of Tripathy's article discusses the solution algorithm used to obtain integer estimates of the variables, something handled automatically by modern linear programming packages.

The Timetabling Problem

The problem dealt with here consists of timetabling a predetermined set of course sections into a section cycle pattern for a given timetabling period. This definition of the problem assumes that a timetabling infrastructure is already in place, that decisions that cross timetabling periods have already been made, and that certain other necessary ancillary decisions have already been made or will be made after the timetable is complete. These restrictions make the timetabling problem completely cross-sectional. Cross-sectional timetabling must be distinguished from the scheduling problems studied by specialists in operations research, the job shop problem for example. These all involve flows of various types and all have important longitudinal components.

The existing timetabling infrastructure includes the definition of the timetabling period and its placement relative to the calendar and also includes the section cycle pattern. The major cross-period scheduling decision to be made in advance is the set of course sections to be offered during each

timetabling period. Each course section will likely be associated with a particular instructor and may be associated with a particular location. As an alternative, the instructor and location may be assigned to each course section after the timetable has been completed. It is also possible to timetable the instructors and locations as additional variables in the timetabling but this greatly increases the complexity of the process.

In my department, I circulated a questionnaire to all departmental faculty members in November asking them for their ranked preferences for courses that they wished to teach during the academic year beginning the following September. A separate questionnaire asked for recommendations for courses to be taught in extension and by outside instructors. The information returned in the questionnaires along with program requirements and our recent history of course offerings provided input into the construction of a list of proposed course sections broken down by semester to be offered during the following academic year along with the assignment of faculty to course sections. Constructing this list required decisions on the courses to be offered in each semester and the number of sections of each course. This phase is done by hand because the scope for real decision making has been narrowed considerably by faculty shortages and by tight upper and lower limits on faculty course load. After circulating the proposed list including faculty assignments to all faculty members and receiving feedback, a final list was constructed. At this point I had in hand the course sections for each timetabling period and the names of the faculty members associated with each section. A second questionnaire was then circulated asking faculty members for preferences with respect to day and time for the courses to which they had been assigned. The next step, the job of the linear programming procedure, was to assign these course sections to section class patterns (time slots) within the cycle pattern. Room assignments were made at the building level after timetabling had been completed.

I digress here to highlight some important differences between the above sequence of events and that described in timetabling textbooks that deal with the secondary level. The differences arise largely from the fact that both students and faculty are much more intensively timetabled at the secondary level, that is, they spend a larger percentage of their timetabled time in class than do post secondary students and faculty. Other differences include the much smaller allowable class size variation at the secondary level and the practice in some secondary schools of keeping student groups intact for all classes. As a consequence, secondary students generally choose their courses for the following year before the timetable is constructed. The detailed information contained in these choices constitutes an important input into the timetabling process and greatly facilitates what is, in essence,

a much more tightly constrained timetabling problem. At the high school level, decisions on the number of course sections to be offered can be based on an accurate count of student numbers. Contrast this with the decision at the university level which is generally based on past experience.

Textbooks on timetabling at the secondary level devote considerable space to working with the conflict matrix, a structure that is not even possible without knowing student course choices in advance. The conflict matrix, whether on paper or in computer memory, contains all courses (but not sections) for the timetabling period in both the rows and columns. Each element of the matrix consists of the number of students who have opted to take both the course in the row and the course in the column. This is the number of potential time conflicts since actual conflicts will not be known until the timetable is complete. It is the number of students who would be prevented from taking one of the two courses if they were timetabled in single sections with the same section class pattern, that is, at the same times on the same days. The diagonal of the constraint matrix, where the row course and the column course are the same, contains the total number of students who have included that course in their selections. The high school timetabler begins with the diagonal to determine the number of required sections for each course. These sections are then timetabled in a manner which minimizes the number of actual time conflicts.

Colleges and universities generally do not have the information required to construct a conflict matrix at the point at which it would be useful, nor do they really need one since the looser post secondary timetabling problem is much less vulnerable to such conflicts. Instead conflicts are anticipated based on program requirements and historical patterns of course choice rather than on current choices. For example, two courses required in the same year of the same program would not be timetabled in the same section pattern in single sections. This conflict problem is explicitly handled by constraints in a linear programming model.

Linear Programming

This is the point at which linear programming is employed to assign course sections to section patterns, or time slots. All practical linear programs must be solved on a computer and most are solved by large mainframe computers. Packages are available to do linear programming on micro computers but these are restricted in the number of variables and constraints that they can handle. I used a mainframe package from IBM that has been available in various revisions for more than 20 years, the "Mathematical Programming System (extended)", usually referred to as "MPSX" or just

"MPS". The input format for MPS has become a standard (Gregory, 1993) that can be accepted by some other packages as well. One of these other packages is IBM's recent successor to MPSX, the Optimization Subroutine Library (1992) or OSL, which is available on computer workstations as well as on mainframe computers.

The complete process involves data preparation for the linear programming input and the preparation of schedules and lists from the output of the linear programming step. A linear program consists of at least one objective function and any number of constraints, all expressed as linear functions of variables. Preparation of the input is done in three phases, variable construction, constraint coding, and objective function specification.

Variable Construction

The structure of the linear programming variables provides considerable insight into the sorts of decisions that the computer makes. From among all variables entered into the program the computer must select a subset, usually a small subset, that contains variables that are both feasible and optimal. This set of basic variables, as they are called, is used to construct timetables and course lists. The variable names used in this exercise consisted of 13 characters. Many computer codes constrain the first character of a variable to be alphabetic and either do not allow lower case characters or allow lower case but ignore case differences. These limitations are observed in the variable definitions below. The variables are prepared conversationally by a program that prompts the user for information in a logical order. This process is described in the next paragraph and the structure of the resulting variables in Table One.

The variable information is prepared one course at a time. The user is initially presented with a table of prefixes and asked to enter the one character corresponding to the course prefix. The course number is then entered. This information appears in characters seven to eleven of the variable name. The computer then requests that the allowable or not allowable class cycle patterns be selected from a menu of such patterns and codes this as the first character of the variable name. This is followed by an elaborate set of questions to specify starting times and period lengths. Times that correspond exactly to the class cycle pattern can be specified quickly as can single offering times. Multiple non-standard times must be specified in detail. This information requires the five columns from two to six. Although there is some provision for non-standard times, the structure of the variable name limits these. For example, since there is provision for only one starting time in a variable name, a course section could not be offered with classes at

Table One

Informational Structure of Linear Programming Variable Names

Character position in variable name	Type of information coded into variable at given position
1	Section class pattern: an arbitrary alphabetic character for each pattern.
2	Starting hour: a numeral or letter for the hour during which classes begin using the 24 hour clock and assigning hours of ten and above letters beginning with "A" and continuing on.
3	Starting tens of minutes: a numeral from zero to five.
4	Starting minute: any single digit numeral but usually zero or five.
5,6	Class period length: nominal period length in units of five minutes.
7	Course prefix: an arbitrary character to represent the course prefix. Although prefixes may be two to four characters long, most institutions have fewer than 36 of these so they can be conveniently represented with one character
8,9,10,11	Course number: four numeric characters at UNB but this can vary with institution.
12,13	Instructor: two characters representing the instructor, usually first and last initials, but arbitrary characters for unknown (to be announced) instructors.

different starting times on different days. If the number of such cases was limited and clearly specified, for example, Friday evening and Saturday morning at specific times, it could be built into the section class patterns. But the practice in some junior high schools of having classes of a given course section start at a variety of times on different days would require a more elaborate structure of the variable name for the section class patterns, the starting times, and perhaps the period lengths. It might even be necessary to timetable the individual class periods rather than the course sections. Finally the program prompts the user for the content of the last two columns of the variable name, the codes for faculty members who can teach the course. These are presented as a menu. At this point a set of variables has been prepared for one course with every allowable combination of section class pattern, starting time, period length, and instructor. The cycle is repeated for the other courses.

The end result for a department of 13 faculty members is a set of 200 variables of which about 35 will be selected by linear programming to form the basis or set of feasible and optimal section offerings for the timetabling period. The 200 variables represent all course section offerings that the timetabler has judged to be possible. The linear programming problem can be simplified by eliminating those course and pattern combinations that are clearly infeasible. For example, some graduate courses must be offered in the evening and so no variables would be coded to represent daytime offerings of such courses. Constraints could be coded to prevent these courses from being offered during the daytime but this needlessly complicates the linear program.

The output of the linear programming step consists of a list of all input variables and their values. Variables not in the basis receive a value of zero, to indicate that the course section designated by the given combination of course, pattern, time, and instructor is not to be offered. Variables in the basis have a value of one to indicate that the section will be offered. Variable values greater than one are not reasonable since this would imply that one instructor could teach multiple sections of a course in a single time slot.

Constraint Coding

Constraints are linear combinations of the variables that are set to be equal to, greater than or equal to, or less than or equal to a constant. For the purpose of timetabling, constraints can be classified as implicit or explicit. Implicit constraints are those that can be derived from the variable names without any additional information. Once the variables are construct-

ed, the complete set of implicit constraints can be generated by a computer program written for that purpose. The set of constraints that prevent a faculty member from teaching multiple course sections at the same time can be derived from the variables since each variable contains the codes giving the time of a course offering and identifying the responsible faculty member. Explicit constraints require the input of additional information before they can be coded. For example, the constraints that prevent sections of different courses from being offered at the same time require information on which pairs of courses must be offered at different times.

Any variable that is infeasible independently of all other variables, if it has been coded in the first place, should be deleted from the variable set rather than constrained. This practice reduces the number of variables and helps to keep the program manageable. Variables that are clearly sub-optimal can also be deleted provided that this does not introduce infeasibilities into the program.

All constraints were coded by dedicated computer programs using as input the set of variables and the other required information for the explicit constraints. For the explicit constraints some of these programs were conversational while others required input in the form of a computer file. The coding of constraint names will not be considered in detail here because this varied according to the type of constraint and constraint names were not used after the linear program had run. It should be noted though that all constraint names began with a single alphabetic character giving the type of constraint according to the detailed classification in the following paragraphs. Each constraint type was only one of greater than or equal to, less than or equal to, or equal to; and this could therefore be determined from the first character of the name. The last character in every constraint name was an integer giving the value of the right hand side of that constraint. The right hand side is the limiting value for the constraint. For example, the right hand side of an equality constraint requiring that a faculty member teach three course sections in a semester would be three.

Implicit Constraints

The first set of implicit constraints is that which prevents any faculty member being assigned two or more course sections with overlapping times. This set of constraints is prepared by a program that examines the variable list, locates course sections taught by the same instructor with conflicting times, and prepares a set of less than or equal to constraints for each potential time conflict for each instructor that limits the maximum number of such course sections to one.

The second set of implicit constraints prevents two sections of the same course from being offered at the same time. It should be pointed out that not every timetable requires this type of constraint. Indeed in some cases, multiple sections of the same course are deliberately set at the same time. In our case these courses were compulsory and students taking courses in different departments needed the extra flexibility provided by different times for different sections.

Explicit Constraints

The first set of explicit constraints controls the number of course sections per instructor. Separate minima and maxima can be coded as greater than or equal to and less than or equal to constraints if the exact number is not known. Since the exact number was known, equality constraints were used to set the number of course sections per instructor to a fixed predetermined number for each instructor.

The second set of explicit constraints controls the number of sections per course. This can be any of the three types of constraints, but since the exact number was known in advance of timetabling an equality constraint was coded for each course to set the number of sections. In most cases this was one, but for a few compulsory undergraduate courses it was greater.

The third set of explicit constraints prevents sections of designated different courses from being offered at the same time. It is the existence of this constraint which, more than any other, complicates timetabling and creates the need for the use of computers. The extra input that makes this an explicit constraint consists of conflict lines. A conflict line is a series of course numbers. In each line, no course after the first can be offered at the same time as the first. The lines are listed in ascending numerical order of the first course and courses in each line after the first are listed in ascending numerical order. This ordering prevents redundant constraints from being coded, constraints that would simply reverse the course order of previous constraints. Generating conflict lines is non-trivial. These were developed in conjunction with departmental faculty and went through several drafts before finalization. Contrast this with the conflict matrices used in programs where students register before courses are timetabled. Obviously, courses that are compulsory in a program in the same term cannot be offered at the same time but there are other restrictions that are highly desirable but not essential. In any given program, options should be offered at different times if possible to increase student choice. Indeed if there is too much concurrent offering of options, students may be unable to register for a complete program. Students may be required to take some options within their major

area but often have the right to take some options in other areas. This is further complicated by the fact that some students may already have taken some of the available options. Another complication is that course conflicts may occur for non-programmatic reasons. Two otherwise non-conflicting courses may require the use of the same shop or laboratory, for example. On the other hand, it is not possible to offer all courses at different times. Part of the problem with course conflicts is to judge how far to go. Too many course conflict constraints could lead to a linear program with no feasible solution or such a small set of feasible solutions that the optimal solution would depart radically from true optimality. Too few could limit student choice and even prevent students from taking complete programs. A course conflict constraint limits the sum of two potentially conflicting course sections to be less than or equal to one.

Objective Function Specification

The objective function is the linear combination of the variables that is maximized to determine, within the set of feasible variable combinations, which particular combination is optimal. The need for a feasible timetable is obvious but what constitutes optimality is less clear. I started with an arbitrary constant for every objective function coefficient and then introduced a stochastic component to eliminate large numbers of identical coefficients which could lead to multiple optimal solutions. A systematic component was then added to each coefficient. For the systematic component, I relied largely on the preferences of faculty members obtained by questionnaire. Most preferences were of the day and time type and these were easy to build into the objective function by assigning higher coefficients to variables representing preferred days and times. The magnitudes of the coefficients involved some judgement, especially where the preferences of different persons were in conflict. I also used the objective function in second and later runs of the program to prevent the undesirable outcomes of earlier runs if a constraint threatened infeasibility.

In the MPS format, the objective function is coded in the same manner as a constraint but is designated as unconstrained whereas a constraint is designated as equal to, greater than or equal to, or less than or equal to.

Input File Preparation

The MPS input format consists of four sections: rows, columns, right hand side, and bounds. The rows section contains the name of each constraint and the objective function and a code for each one to indicate the type

of constraint or, in the case of the objective function, that it is unconstrained. The columns section contains the names of all variables associated with each constraint name or objective function for which the coefficient is non-zero, and the value of that coefficient. The various constraint generating programs produce the columns section of the MPS input. The right hand side section includes all constraint names and their limiting values. The bounds section contains upper or lower limits for any bounded variables. All variables, as defined here, have an upper bound of one. In this particular formulation of the timetabling problem, the entire MPS input file can be generated from the columns section and this is done by a dedicated program as the last step before the linear program is run. Since the columns section contains the names of all constraints and the name of each constraint identifies its type and contains its limiting value, the rows and right hand side sections can be derived from the columns section. For the bounds section, all variable names are extracted from the columns section and are set to an upper limit of one.

The input file format also allows variables to be restricted to integer values. In the linear programming examples reviewed above this was done since fractional course sections are never taught. Linear programming packages that can do integer programming generally solve the problem first without restricting the variables to integers and then impose the integer restriction on those variables that are not already integers. The difficulty is that integer linear programming is much more time consuming than general linear programming. In this example, the variables were lower-bounded at zero by the non-negativity constraints built into linear programming packages and upper-bounded at one by the bounds section. Without an integer restriction, they could still assume fractional values between zero and one. An integer restriction was imposed on all variables even though, in practice, nearly all of the variables assumed integer values without the need for integer linear programming. This seemed to be a consequence of the fact that all variables were constrained by several different sets of equality constraints with integer values on the right hand side. The important point here is that very little extra computer time was required to force the variables to integer values.

A university department of 13 faculty members offering 30 to 40 course sections per semester generates a linear program of 200 variables, 380 constraints, and 1400 non-zero elements in the objective function and the constraints. Increasing the size of the department would increase the number of variables proportionately but would likely increase the number of constraints more than proportionately. The increase in the size of the constraint set would depend not only on the increase in the numbers of faculty members and course sections offered but also on any change in the

complexity of the program offerings. Decomposition techniques can be used with subprograms that are almost independent to split the program into a number of smaller programs while still allowing the co-ordination required by a few common courses or instructors. A large number of academic programs with numerous or complex cross-linkages would produce a large and cumbersome linear program.

Ad Hoc Adjustments

Completion of the timetabling phase involved multiple runs of the linear program. After some runs, timetables and lists were prepared and circulated to faculty members for comment. The comments were used as input into the subsequent runs of the program. Other ad hoc adjustments had to be made because some desirable characteristics of the timetable could not be represented easily in a linear programming format. The ad hoc adjustments were made to the columns section before the complete input file was prepared. The constraint generating programs that produced the columns section added an explanatory comment to the end of each line that facilitated the alteration of the columns section with a text editor.

A requirement that is impossible to code in a linear program, but one that I imposed, is that each faculty member have one weekday with no classes. To achieve this, I ran the program without this constraint and then chose an appropriate weekday off for each faculty member based on the initial assignment of course sections. For many faculty members, the initial run had generated one or more weekdays without classes. I ran the program again after either lowering the objective function coefficients for the chosen day off or deleting all the variables corresponding to course sections on the day off.

Some adjustments, while not necessary, simplified the linear program. Individual faculty members who preferred to teach at times that were unpopular with the majority of faculty members had the variables representing their course sections at other times deleted. This ensured that they got their preferences and reduced the number of variables and constraints in the linear program.

Linear Programming Output

The output from The MPS system contains all variables and their values. The same job that generated the linear programming solution contained a step that filtered the output to eliminate all variables with zero values. What remained were the variables with values of one representing

all course sections chosen as the most optimal of the feasible solutions. These were then used to produce course lists and timetables.

Typically, three to five runs were required with the circulation of two or three draft timetables to faculty members before a near final timetable was obtained. The final adjustments were often required by unforseen circumstances occurring after it had become impossible to make major changes. These changes were made by hand by altering the list of optimal variables to eliminate or add a course section or to change an instructor or some other aspect of a course. New lists and timetables could then be generated from the revised set of optimal variables.

The results were generally well received by faculty members. Problems occasionally arose when the file of conflict lines omitted courses that should have been included and when faculty members neglected to indicate strongly held preferences on the preference questionnaire. The resulting timetables can be considered optimal to the extent that we were able to define optimality.

Application to Conflict Matrices

Some interesting twists arise when applying these techniques to the high school situation with course registration occurring before timetabling. The aim of the high school timetabler is to minimize course conflicts but the assumption is often made that they cannot be eliminated entirely. In linear programming terminology, there is no feasible solution when conflicts cannot be eliminated. But it is difficult to accept that the additional information in the hands of the high school timetabler makes timetabling more rather than less difficult. The difference is that university students are forced to choose non-conflicting courses from a timetable regardless of their preferences whereas high school students eventually receive a timetable that reflects their preferences.

The high school timetabler might be tempted to build information from the conflict matrix into the objective function to make conflicts suboptimal rather than infeasible but this turns out to be impossible. Negative weights could be placed on two course sections with a large number of conflicts except that the weighting coefficients would need to multiply the product of the two course section variables. This is neither linear nor separable and cannot be entered into or solved by a linear program.

A more viable approach would be to code conflict lines for all course combinations that have non-zero entries in the conflict matrix. If the linear

programming procedure did not reach a feasible solution with this constraint set, constraints could be removed successively for course combinations for which the constraint matrix entries were the lowest until a feasible solution was reached. In fact, the timetabler could make judgements about the undesirability of a conflict based on information additional to that contained in the conflict matrix and remove the constraints in a different order.

Conclusions

Linear programming is an effective and viable means of timetabling at the university level. It alone among timetabling techniques goes beyond finding a merely feasible solution and locates, among the set of feasible timetables, one that is optimal. Unfortunately, running linear programs large enough to do non-trivial timetabling requires a mainframe computer or, at least, a workstation. This will likely change as the line between microcomputers and workstations becomes blurred as a result of the more powerful microprocessors and more sophisticated operating systems becoming available for microcomputers. It can now be said that earlier concerns about excessive amounts of computer time being required for timetabling with linear programming no longer have any validity. A real disadvantage is the fact that linear programming packages are not particularly easy to use and generally require specialized knowledge of the user. This could be at least partly avoided by a user friendly front end specifically aimed at university timetabling. But such a front end does not now exist and the user is forced to resort to custom programming. Nevertheless linear programming may have a future in timetabling since timetabling is essentially linear and there are real advantages to being able to locate optimal solutions.

Bibliography

Abramson, D. (1991 January). "Constructing School Timetables Using Simulated Annealing: Sequential and Parallel Algorithms". *Management Science*. 37,1: 98-113.

Aubin, Jean and Ferland, Jacques A. (1989). "A Large Scale Timetable Problem". *Computers and Operations Research*. 16,1: 66-77.

Brookes, John E. (1980). *Timetable Planning*. London, U.K.: Heinemann Educational Books. 114 pp. (LB 1038 B76).

Carlson, R.C. and Nemhauser, G.L. (1966 January-February). "Scheduling to Minimize Interaction Cost". *Operations Research*. 14,1: 52-58.

Chapman, William Dan and Gambrell, C.B. (1976 Winter). "G.R.A.S.S is Available". *College and University*. 51,2: 238-257.

Dantzig, George B. (1963). *Linear Programming and Extensions*. Princeton, New Jersey: Princeton University Press. 632 pp.

Delacour, A.W. (1971). *Logical Timetabling*. Porthcawl, U.K.: Pulin Publishing. 128 pp. (LB 1038 D44).

Dempsey, Richard A. and Traverso, Henry P. (1983). *Scheduling the Secondary School*. Reston, Virginia: National Association of Secondary School Principals. 82 pp. (LB 1038 .D46 1983).

Folkers, J.S. (1969). "Research and Management Aspects of Time-table automation for Schools and Universities". *Efficiency in Resource Utilization in Education*. Paris, France: Organization for Economic Co-operation and Development. 217-241. (LB 2824 .O7).

Gass, Saul I. (1970). *An Illustrated Guide to Linear Programming*. New York: Dover Publications Inc. 173 pp.

Gregory, John W. (jwg@cray.com). (1993 03 24). *Linear Programming: Frequently Asked Questions List*. garbo@uwasa.fi: /pc/math /faqlp303.zip.

Haley, K.B. (1962 July-August). "The Solid Transportation Problem". *Operations Research*. 10,4: 448-463.

Introduction to Mathematical Programming System — Extended (MPSX), Mixed Integer Programming (MIP) and Generalized Upper Bounding (GUB) (4 ed) (IBM GH20-0849-3). (1973 September). White Plains, New York: International Business Machines Inc. 37 pp.

King, Horace C. and Martin, Frank B. (1976 Winter). "Computer Eases Student Scheduling Problem". *College and University*. 51,2: 165-173.

Kolman, Bernard and Beck, Robert E. (1980). *Elementary Linear Programming with Applications* Orlando, Florida: Academic Press Inc. 399 pp.

LaTourrette, Guy and Klee, Jean. (1970). *Programmation linéaire et ordinateur (Initiation pratique)*. Paris: Entreprise Moderne d'Edition. 129 pp.

Lawrie, N.L. (1969). "An integer linear programming model of a school timetabling problem". *Computer Journal*. 12: 307-316.

Lions, John. (1969). "The Construction of Timetables for Ontario Schools Using a Computer". *Efficiency in Resource Utilization in Education*. Paris, France: Organization for Economic Co-operation and Development. 147-171. (LB 2824 .O7).

Mallett, Mark. (1991 12). "Sorting Out Schedules". *Byte*. 16,13: 263-264,266,268.

McKillop, Rufus E. (1978 April). *An Investigation of the University Timetabling Problem* (Master of Science thesis). Fredericton: University of New Brunswick.

Morávek, J. and Vlach M. (1967 May-June). "On the Necessary Conditions for the Existence of the Solution of the Multi-index Transportation Problem". *Operations Research*. 15,3: 542-546.

Oppenheimer, Kenneth R. (1978 02). "A Proxy Approach to Multi-attribute Decision Making". *Management Science*. 24,6: 675-689.

Optimization Subroutine Library Guide and Reference (Release 2, 4ed) (IBM SC23-0519-3). (1992 June). Kingston, New York: International Business Machines Inc. 823 pp.

"Scheduling: From Micro to Macro". (1991 October). *The Practitioner*. Reston, Virginia: National Association of Secondary School Principals. 18,1: 1-8.

Schmidt, G. and Ströhlein T. (1980 November). "Timetable Construction—an annotated bibliography". *Computer Journal*. 23,4: 307-316.

School Timetabling (Unit 9 in Educational Studies: A third level course). (1976). Milton Keynes, U.K.: Open University Press. 40 pp.

Selim, S.M. (1982). "An Algorithm for Constructing a University Faculty Timetable". *Computers and Education*. 6,4: 323-332.

Simper, Roger. (1980). *A Practical Guide to Timetabling*. London, U.K.: Ward Lock Educational. 147 pp.

Tripathy, Arabinda. (1984 December). "School Timetabling—A Case in Large Binary Integer Linear Programming". *Management Science*. 30,12: 1473-1489.

Walton, Jack (ed). (1972). *The Secondary School Timetable*. London, U.K.: Ward Lock Educational. 136 pp. (LB 3033 .W357).